**Министерство образования Республики Беларусь**

**УО “Барановичский государственный университет”**

Инженерный факультет

Кафедра информационных технологий

И физико-математических дисциплин (ИСТ)

Средства визуальной разработки программных приложений

Лабораторная работа № 5

Вариант № 4

группа ИСТ-41

Ф.И.О. студента Борисов И.А.

Ф.И.О. преподавателя Шапович Е. Г.

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барановичи, 2023

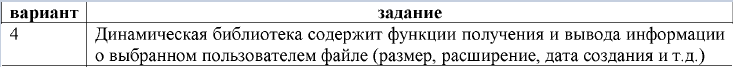
**Лабораторная работа 5**

**Создание и использование библиотек DLL**

**Цель**: Овладение инструментальными средствами и методами по использованию dll.

**Задание**

Разработать dll библиотеку обладающую функционалом в соответствии с заданием и приложение демонстрирующее работу созданной библиотеки.



**Код программы:**

DllEntry.cpp

BOOL APIENTRY DllMain(HMODULE hModule,

DWORD ul\_reason\_for\_call,

LPVOID lpReserved

)

{

switch (ul\_reason\_for\_call)

{

case DLL\_PROCESS\_ATTACH:

case DLL\_THREAD\_ATTACH:

case DLL\_THREAD\_DETACH:

case DLL\_PROCESS\_DETACH:

break;

}

return TRUE;

}

std::wstring str2wstr(const std::string &str)

{

int size\_fileCreated = MultiByteToWideChar(CP\_UTF8, 0, &str[0], (int)str.size(), NULL, 0);

std::wstring wstrTo(size\_fileCreated, 0);

MultiByteToWideChar(CP\_UTF8, 0, &str[0], (int)str.size(), &wstrTo[0], size\_fileCreated);

return wstrTo;

}

// Export function

extern "C" \_\_declspec(dllexport) int getInfo(HWND hwnd, UINT msg, WPARAM wParam,

LPARAM lParam, HWND textMemo)

{

OPENFILENAME openFile;

char sizeFileName[MAX\_PATH] = "";

FILETIME ft\_create, // Creation Time

ft\_access, // Access time

ft\_write; // Write time

SYSTEMTIME syst\_create, syst\_lastWrite;

char str\_creationTime[100], str\_lastWrite[100];

std::wstring

wstr\_fileAttInfo,

wstr\_fileAttAll,

wstr\_attReadOnly,

wstr\_attHidden,

wstr\_attDirectory,

wstr\_attArchive,

wstr\_attNormal,

wstr\_fileSize,

wstr\_fileSizeFind,

wstr\_fileCreated,

wstr\_fileChanged,

wstr\_fileOwner,

wstr\_fileInformation,

wstr\_fileSizeAndDate;

ZeroMemory(&openFile, sizeof(openFile));

openFile.lStructSize = sizeof(OPENFILENAME);

openFile.lpstrTitle = L"Select file";

openFile.hwndOwner = hwnd;

openFile.nFilterIndex = 2;

openFile.lpstrFilter = L"User files (\*.usf)\0\*.usf\0All files (\*.\*)\0\*.\*\0";

openFile.lpstrFile = (LPWSTR)sizeFileName;

openFile.nMaxFile = sizeof(sizeFileName);

openFile.lpstrInitialDir = NULL;

openFile.Flags = OFN\_EXPLORER | OFN\_FILEMUSTEXIST | OFN\_HIDEREADONLY | OFN\_PATHMUSTEXIST;

// Get info about file

if (GetOpenFileName(&openFile) == TRUE)

{

// Get file attributes

DWORD fileAttributes = GetFileAttributes(openFile.lpstrFile);

//hFile = CreateFile(openFile.lpstrFile, GENERIC\_READ, FILE\_SHARE\_READ, NULL,

// OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (fileAttributes != INVALID\_FILE\_ATTRIBUTES)

{

// **Get file name**

std::wstring wstr\_fileName(openFile.lpstrFile);

// **Get file extension**

std::wstring wstr\_extension = wstr\_fileName.substr(wstr\_fileName.find\_last\_of(L".") + 1);

HANDLE fileHandle = CreateFile(openFile.lpstrFile, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (fileHandle != INVALID\_HANDLE\_VALUE)

{

LARGE\_INTEGER fileSize;

char str\_fileSizeAndDate[1000];

//**Get file time**

GetFileTime(fileHandle, &ft\_create, &ft\_access, &ft\_write);

// **Creation Time conversion**

FileTimeToSystemTime(&ft\_create, &syst\_create);

GetDateFormatA(LOCALE\_USER\_DEFAULT, 0, &syst\_create, "yyyy-MM-dd", str\_creationTime, 100);

GetTimeFormatA(LOCALE\_USER\_DEFAULT, 0, &syst\_create, "HH:mm:ss", str\_creationTime + 10, 100 - 10);

// **Last write Time conversion**

FileTimeToSystemTime(&ft\_write, &syst\_lastWrite);

GetDateFormatA(LOCALE\_USER\_DEFAULT, 0, &syst\_lastWrite, "yyyy-MM-dd", str\_lastWrite, 100);

GetTimeFormatA(LOCALE\_USER\_DEFAULT, 0, &syst\_lastWrite, "HH:mm:ss", str\_lastWrite + 10, 100 - 10);

**// Get file size**

GetFileSizeEx(fileHandle, &fileSize);

sprintf\_s(str\_fileSizeAndDate, "File creation date: %s\r\nFile change date: %s\r\nSize of file: %lld bytes", str\_creationTime,

str\_lastWrite, fileSize.QuadPart);

// Char to conversion

wstr\_fileSizeAndDate = str2wstr(str\_fileSizeAndDate);

CloseHandle(fileHandle);

}

else { wstr\_fileSizeAndDate = L"Error, not open file"; }

// File owner

DWORD buffer\_size = GetFileSecurity(openFile.lpstrFile, OWNER\_SECURITY\_INFORMATION, NULL, 0, &buffer\_size);

if (buffer\_size > 0)

{

PSECURITY\_DESCRIPTOR securityDescriptor = reinterpret\_cast<PSECURITY\_DESCRIPTOR>(malloc(buffer\_size));

if (GetFileSecurity(openFile.lpstrFile, OWNER\_SECURITY\_INFORMATION, securityDescriptor,

buffer\_size, &buffer\_size))

{

PSID owner\_Sid = NULL;

BOOL owner\_defaulted = FALSE;

if (GetSecurityDescriptorOwner(securityDescriptor, &owner\_Sid, &owner\_defaulted))

{

TCHAR owner\_name[MAX\_PATH] = { 0 };

DWORD owner\_nameSize = sizeof(owner\_name);

SID\_NAME\_USE sid\_nameUse;

if (LookupAccountSid(NULL, owner\_Sid, owner\_name, &owner\_nameSize,

NULL, NULL, &sid\_nameUse))

{

wstr\_fileOwner = static\_cast<std::wstring> (owner\_name);

}

}

}

free(securityDescriptor);

}

**// Path to folder**

std::wstring wstr\_fileFolderPath = wstr\_fileName.substr(0, wstr\_fileName.find\_last\_of(L"\\"));

// For

wstr\_fileInformation = L"File name: " + wstr\_fileName + L"\r\n";

wstr\_fileInformation += L"File extension: " + wstr\_extension + L"\r\n";

wstr\_fileInformation += L"Folder path: " + wstr\_fileFolderPath + L"\r\n";

wstr\_fileInformation += wstr\_fileSizeAndDate + L"\r\n";

//wstr\_fileInformation += L"File change date: " + wstr\_fileChanged + L"\r\n";

// wstr\_fileInformation += L"File owner: " + wstr\_fileOwner + L"\r\n";

// Clear WC\_EDIT -> Memo

SetWindowTextW(textMemo, L"");

// Output info -> Memo

SendMessageW(textMemo, EM\_REPLACESEL, FALSE, (LPARAM)wstr\_fileInformation.c\_str());}}

else

{

// std::cout << "File selection canceled!\n";

MessageBox(hwnd, L"File selection canceled!\nCould not get information about the file",

L"Error", MB\_OK | MB\_ICONERROR);

}

return 0;}

MainGUIWindow.cpp

//MainForm

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdShow)

{

WNDCLASS MainGuiClass = MainWindowClass((HBRUSH)COLOR\_WINDOW, LoadCursor(NULL, IDC\_ARROW), hInst, LoadIcon(NULL, IDI\_WINLOGO),

L"MainWindowClass", MainWindowProcedure);

if (!RegisterClassW(&MainGuiClass)) { return -1; }

MSG MainGuiMessage = { 0 };

CreateWindow(L"MainWindowClass", L"Info Application", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 300, 300, 600, 450, NULL, NULL, NULL, NULL);

while (GetMessage(&MainGuiMessage, NULL, NULL, NULL))

{

TranslateMessage(&MainGuiMessage);

DispatchMessage(&MainGuiMessage);

}

UnregisterClass(MainGuiClass.lpszClassName, hInst);

return 0;

}

//MainFormClass

WNDCLASS MainWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure)

{

WNDCLASS MainWindowClass = { 0 };

MainWindowClass.hIcon = Icon;

MainWindowClass.hCursor = Cursor;

MainWindowClass.hIcon = Icon;

MainWindowClass.lpszClassName = Name;

MainWindowClass.hbrBackground = BGColor;

MainWindowClass.lpfnWndProc = Procedure;

return MainWindowClass;

}

// For class

//typedef Informator \*(\*CreateInformatorClass)();

//typedef void (\*DeleteInformatorClass)(Informator \*);

// Function prototype from DLL

typedef int(\_\_cdecl \*GetInfoFunc)(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam, HWND memo);

HRESULT CALLBACK MainWindowProcedure(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

HINSTANCE hDLL = LoadLibrary(L"DLL5.dll");

if (hDLL == NULL) {

MessageBox(hwnd, L"DLL Error", L"Error", MB\_OK | MB\_ICONERROR); return 1;

}

switch (message)

{

case WM\_COMMAND:

if (LOWORD(wParam) == OnBtSelectFileClicked)

{

MainGuiSelectFileClicked(hDLL, hwnd, message, wParam, lParam, textboxMemo);

}

break;

case WM\_CREATE:

MainGuiWidgets(hwnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default: return DefWindowProc(hwnd, message, wParam, lParam);}}

//Widgets - add controls

void MainGuiWidgets(HWND hWnd)

{

//labels

labelX1 = CreateWindowEx(0, WC\_STATIC, L"Output", WS\_CHILD | WS\_VISIBLE, 12, 20, 820, 130, hWnd, nullptr, nullptr, nullptr);

// Button 'Select file'

HWND button = CreateWindow(L"BUTTON",

L"Select file",

WS\_BORDER | WS\_VISIBLE | WS\_CHILD,

125, 370, 250, 24,

hWnd,

(HMENU)OnBtSelectFileClicked,

NULL,

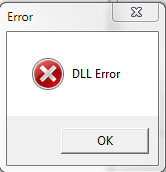
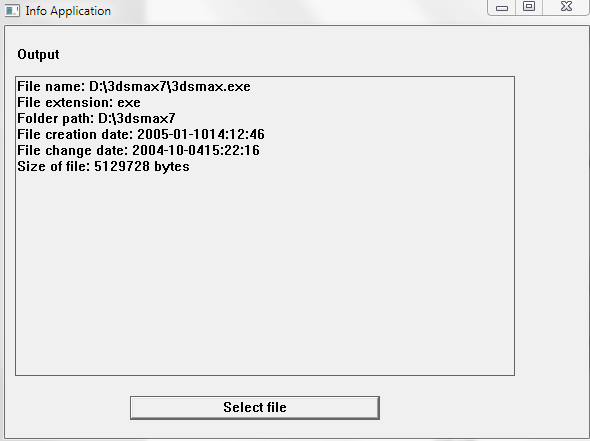
NULL);

// WC\_EDIT Memo

textboxMemo = CreateWindowEx(0, L"EDIT", NULL, WS\_CHILD | WS\_VISIBLE | WS\_BORDER | ES\_MULTILINE | ES\_AUTOVSCROLL | ES\_READONLY,

10, 50, 500, 300, hWnd, NULL, NULL, NULL);}

**Результат программы:**



**Вывод**: овладел инструментальными средствами и методами по созданию и использованию dll библиотек в графическом приложении.